**AWS SSM Deployment Guide**

HTML to DOCX/Confluence Converter

# Overview

This guide covers deploying the HTML to DOCX/Confluence converter on AWS using Systems Manager (SSM). There are two primary deployment methods:

1. **EC2 Instance Method** - Deploy directly on EC2 instances managed by SSM
2. **Docker Container Method** - Run as a containerized task via SSM

**Important:** Both methods require headless Chromium to capture interactive charts, which necessitates proper system dependencies.

# Method 1: EC2 Instance Deployment

This method installs all dependencies directly on EC2 instances that are managed by AWS Systems Manager.

## Prerequisites

* EC2 instances with SSM agent installed and running
* IAM role attached to instances with SSM and S3 permissions
* Amazon Linux 2 or Ubuntu 20.04+ operating system
* Network access to install packages

## Step 1: Create IAM Role for EC2

Create an IAM role with the following managed policies:

* AmazonSSMManagedInstanceCore - Required for SSM management
* AmazonS3ReadOnlyAccess - Required to read HTML from S3

**Custom policy for Confluence (optional):**

{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"secretsmanager:GetSecretValue"
],
"Resource": "arn:aws:secretsmanager:\*:\*:secret:confluence/\*"
}
]
}

## Step 2: Launch EC2 Instance

Launch an EC2 instance with the following configuration:

* **Instance Type:** t3.medium or larger (Chromium requires memory)
* **AMI:** Amazon Linux 2 or Ubuntu 20.04+
* **IAM Role:** Attach the role created in Step 1
* **Storage:** 20 GB minimum (for Chromium and dependencies)
* **Tags:** Add Environment and Application tags for SSM targeting

## Step 3: Create Bootstrap Script

Create a user data script or run manually via SSM to install dependencies:

**For Amazon Linux 2:**

#!/bin/bash
# Update system
sudo yum update -y
# Install Python 3 and pip
sudo yum install -y python3 python3-pip
# Install system dependencies for Chromium
sudo yum install -y \
alsa-lib atk cups-libs gtk3 \
ipa-gothic-fonts libXcomposite \
libXcursor libXdamage libXext \
libXi libXrandr libXScrnSaver \
libXtst pango xorg-x11-fonts-100dpi \
xorg-x11-fonts-75dpi xorg-x11-fonts-cyrillic \
xorg-x11-fonts-misc xorg-x11-fonts-Type1 \
xorg-x11-utils
# Install Python packages
sudo pip3 install playwright beautifulsoup4 \
html2docx boto3 atlassian-python-api lxml
# Install Chromium for Playwright
sudo -u ec2-user playwright install chromium
sudo playwright install-deps
echo "Bootstrap complete!"

**For Ubuntu 20.04+:**

#!/bin/bash
# Update system
sudo apt-get update
# Install Python 3 and pip
sudo apt-get install -y python3 python3-pip
# Install system dependencies for Chromium
sudo apt-get install -y \
libnss3 libxss1 libasound2 \
libxrandr2 libatk1.0-0 libgtk-3-0 \
libgbm-dev libx11-xcb1 libxcb-dri3-0 \
libdrm2 libgbm1 libegl1
# Install Python packages
sudo pip3 install playwright beautifulsoup4 \
html2docx boto3 atlassian-python-api lxml
# Install Chromium for Playwright
sudo -u ubuntu playwright install chromium
sudo playwright install-deps
echo "Bootstrap complete!"

## Step 4: Deploy the Script

Upload your Python script to S3 or create it directly on the instance:

# Option 1: Copy from S3
aws s3 cp s3://your-bucket/html\_converter.py /opt/html\_converter.py
# Option 2: Create directly
sudo mkdir -p /opt/scripts
sudo vi /opt/scripts/html\_converter.py
# Paste your script content
# Set permissions
sudo chmod +x /opt/scripts/html\_converter.py

## Step 5: Configure Script Settings

Edit the configuration at the top of the script for AWS mode:

# Set to S3 mode
HTML\_SOURCE = "S3"
OUTPUT\_DESTINATION = "Confluence"
# Configure S3 source
S3\_BUCKET = "your-reports-bucket"
S3\_HTML\_KEY = "reports/monthly-report.html"
# Configure Confluence
CONFLUENCE\_URL = "https://your-domain.atlassian.net"
CONFLUENCE\_SPACE\_KEY = "TEAM"
CONFLUENCE\_PARENT\_PAGE\_ID = "123456789"
CONFLUENCE\_PAGE\_TITLE = "Automated Report"

## Step 6: Create SSM Document

Create an SSM document to run the converter:

---
schemaVersion: '2.2'
description: 'Convert HTML from S3 to Confluence'
parameters:
S3Bucket:
type: String
description: 'S3 bucket containing HTML file'
default: 'your-reports-bucket'
S3Key:
type: String
description: 'S3 key for HTML file'
default: 'reports/monthly-report.html'
ConfluenceTitle:
type: String
description: 'Title for Confluence page'
default: 'Automated Report'
mainSteps:
- action: 'aws:runShellScript'
name: 'RunConverter'
inputs:
runCommand:
- '#!/bin/bash'
- 'export CONFLUENCE\_USER="{{ssm:/confluence/user}}"'
- 'export CONFLUENCE\_TOKEN="{{ssm:/confluence/token}}"'
- 'cd /opt/scripts'
- 'python3 html\_converter.py'
- 'echo "Conversion complete"'

## Step 7: Execute via SSM

Run the SSM document:

# Run on specific instance
aws ssm send-command \
--document-name "HTML-to-Confluence" \
--targets "Key=instanceids,Values=i-1234567890abcdef0" \
--parameters "S3Bucket=my-bucket,S3Key=reports/report.html"
# Run on all instances with specific tag
aws ssm send-command \
--document-name "HTML-to-Confluence" \
--targets "Key=tag:Environment,Values=Production" \
--parameters "S3Bucket=my-bucket,S3Key=reports/report.html"

# Method 2: Docker Container Deployment

This method runs the converter in a Docker container, providing better isolation and easier dependency management.

## Prerequisites

* EC2 instance with Docker installed
* Amazon ECR repository for Docker images
* IAM role with ECR, S3, and SSM permissions

## Step 1: Create Dockerfile

Create a Dockerfile for the converter:

FROM mcr.microsoft.com/playwright/python:v1.40.0-jammy
WORKDIR /app
# Copy requirements
COPY requirements.txt .
# Install Python dependencies
RUN pip install --no-cache-dir -r requirements.txt
# Copy the script
COPY html\_converter.py .
# Set environment variables (override at runtime)
ENV HTML\_SOURCE=S3
ENV OUTPUT\_DESTINATION=Confluence
# Run the script
CMD ["python", "html\_converter.py"]

## Step 2: Create requirements.txt

playwright==1.40.0
beautifulsoup4==4.12.2
html2docx==1.6.0
lxml==4.9.3
boto3==1.34.0
atlassian-python-api==3.41.0

## Step 3: Build and Push Docker Image

# Build the image
docker build -t html-converter:latest .
# Tag for ECR
docker tag html-converter:latest \
123456789012.dkr.ecr.us-east-1.amazonaws.com/html-converter:latest
# Login to ECR
aws ecr get-login-password --region us-east-1 | \
docker login --username AWS --password-stdin \
123456789012.dkr.ecr.us-east-1.amazonaws.com
# Push to ECR
docker push \
123456789012.dkr.ecr.us-east-1.amazonaws.com/html-converter:latest

## Step 4: Create SSM Document for Docker

---
schemaVersion: '2.2'
description: 'Run HTML converter in Docker via SSM'
parameters:
ECRImage:
type: String
description: 'ECR image URL'
default: '123456789012.dkr.ecr.us-east-1.amazonaws.com/html-converter:latest'
S3Bucket:
type: String
description: 'S3 bucket'
S3Key:
type: String
description: 'S3 key'
mainSteps:
- action: 'aws:runShellScript'
name: 'RunDockerContainer'
inputs:
runCommand:
- '#!/bin/bash'
- 'echo "Logging into ECR..."'
- 'aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 123456789012.dkr.ecr.us-east-1.amazonaws.com'
- 'echo "Pulling latest image..."'
- 'docker pull {{ECRImage}}'
- 'echo "Running converter..."'
- 'docker run --rm \\'
- ' -e AWS\_DEFAULT\_REGION=us-east-1 \\'
- ' -e S3\_BUCKET={{S3Bucket}} \\'
- ' -e S3\_KEY={{S3Key}} \\'
- ' -e CONFLUENCE\_USER="{{ssm:/confluence/user}}" \\'
- ' -e CONFLUENCE\_TOKEN="{{ssm:/confluence/token}}" \\'
- ' {{ECRImage}}'
- 'echo "Conversion complete"'

## Step 5: Execute Docker Container via SSM

aws ssm send-command \
--document-name "HTML-to-Confluence-Docker" \
--targets "Key=tag:Environment,Values=Production" \
--parameters "S3Bucket=my-bucket,S3Key=reports/report.html"

# Alternative: ECS Fargate (Recommended)

For production workloads, consider using ECS Fargate instead of SSM for better scalability and management.

## Step 1: Create ECS Task Definition

{
"family": "html-converter",
"networkMode": "awsvpc",
"requiresCompatibilities": ["FARGATE"],
"cpu": "1024",
"memory": "2048",
"executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
"taskRoleArn": "arn:aws:iam::123456789012:role/html-converter-task-role",
"containerDefinitions": [
{
"name": "html-converter",
"image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/html-converter:latest",
"essential": true,
"environment": [
{ "name": "HTML\_SOURCE", "value": "S3" },
{ "name": "OUTPUT\_DESTINATION", "value": "Confluence" }
],
"secrets": [
{
"name": "CONFLUENCE\_USER",
"valueFrom": "arn:aws:secretsmanager:us-east-1:123456789012:secret:confluence-user"
},
{
"name": "CONFLUENCE\_TOKEN",
"valueFrom": "arn:aws:secretsmanager:us-east-1:123456789012:secret:confluence-token"
}
],
"logConfiguration": {
"logDriver": "awslogs",
"options": {
"awslogs-group": "/ecs/html-converter",
"awslogs-region": "us-east-1",
"awslogs-stream-prefix": "ecs"
}
}
}
]
}

## Step 2: Trigger via EventBridge

{
"source": ["aws.s3"],
"detail-type": ["Object Created"],
"detail": {
"bucket": { "name": ["my-reports-bucket"] },
"object": { "key": [{ "prefix": "reports/" }] }
}
}

# Troubleshooting

## Common Issues

**Issue: Chromium fails to launch**

Solution: Ensure all system dependencies are installed. Run playwright install-deps as root.

**Issue: Charts not capturing**

Solution: Increase the wait timeout in render\_and\_screenshot() from 1200ms to 2000ms or more.

**Issue: Out of memory errors**

Solution: Use t3.medium or larger instance type. Docker containers need at least 2GB memory allocated.

**Issue: SSM document fails**

Solution: Check SSM agent status with sudo systemctl status amazon-ssm-agent

## Viewing Logs

**SSM Command Output:**

aws ssm get-command-invocation \
--command-id "<command-id>" \
--instance-id "<instance-id>"

**CloudWatch Logs:**

aws logs tail /aws/ssm/html-converter --follow

# Best Practices

## Security

* Store Confluence credentials in AWS Secrets Manager or SSM Parameter Store (SecureString)
* Use IAM roles instead of access keys
* Enable VPC endpoints for S3 and ECR to avoid internet traffic
* Regularly update base Docker images and dependencies

## Performance

* Use t3.medium or larger for EC2 instances
* Allocate at least 2GB memory for Docker containers
* Cache Docker images in ECR close to execution region
* Use S3 Transfer Acceleration for large HTML files

## Monitoring

* Enable CloudWatch Logs for SSM commands
* Set up CloudWatch Alarms for failed executions
* Track execution duration and memory usage
* Monitor S3 bucket for new HTML files

# Cost Optimization

* **EC2 Method:** Use Spot Instances for non-critical workloads (up to 90% savings)
* **Docker Method:** Use Fargate Spot for batch processing
* **Scheduling:** Stop instances when not in use
* **Storage:** Clean up temporary files and old Docker images

**End of Guide**